Attorney Docket No.: 010327-007210US
Client Reference No.: 20

PATENT APPLICATION

METHOD AND SYSTEM FOR IMPLEMENTING EXTERNAL APPLICATIONS USING REMOTE SOCKET APPLICATION PROGRAMMING INTERFACE FOR VIRTUAL ROUTERS

Inventor(s): Bhasker Allam, a citizen of India, residing at
5427 Matthew Terrace
Fremont, CA 94555

Shankar Agarwal, a citizen of India, residing at
1186 Ocaso Camino
Fremont, CA 94539

Assignee: Network Equipment Technologies, Inc.
6900 Paseo Padre Parkway
Fremont, CA, 94555

Entity: Large Business Concern

TOWNSEND and TOWNSEND and CREW LLP
Two Embarcadero Center, Eighth Floor
San Francisco, California 94111-3834
Tel: 415-576-0200

METHOD AND SYSTEM FOR IMPLEMENTING EXTERNAL APPLICATIONS USING REMOTE SOCKET APPLICATION PROGRAMMING INTERFACE FOR VIRTUAL ROUTERS

CROSS-REFERENCES TO RELATED APPLICATIONS

[0001] The present application claims the benefit of priority under 35 U.S.C. § 119 from U.S. Provisional Patent Application Serial No. 60/455,706, entitled "METHOD FOR IMPLEMENTING EXTERNAL APPLICATIONS USING REMOTE SOCKET API FOR VIRTUAL ROUTERS" filed on March 17, 2003, the disclosure of which is hereby incorporated by reference in its entirety for all purposes.

BACKGROUND OF THE INVENTION

[0002] The present invention generally relates to routers and, more specifically, to virtual routers in a split plane architecture.

[0003] A physical router typically includes a number of physical interfaces that are coupled respectively to corresponding packet sources. Packets received from the packet sources are received via the physical interfaces and forwarded by the physical router to their intended destinations.

[0004] A virtual router is generally defined as a collection of threads, either static or dynamic, in a routing device that provides routing and forwarding services similar to those offered by physical routers. A virtual router need not be a separate operating system process. The virtual router simply has to provide the perception or illusion that a dedicated router is available to satisfy the needs of the network(s) to which it is connected. A virtual router, like its physical counterpart, is an element in a routing domain which may include other routers that are either physical or virtual.

[0005] Virtual routers are used to implement Layer-3 virtual private networks (VPNs). A VPN is defined as one or more wide area network (WAN) links over a shared public network, typically over the Internet or an IP (Internet Protocol) backbone from a network service provider that simulates the behavior of dedicated WAN links over leased lines. Virtual routers support VPNs at Layer-3 of the Open Systems Interconnections (OSI) internetworking model. Network links that constitute network ports, VLANs (virtual local area networks) or virtual circuits are partitioned amongst the virtual routers. The virtual

routers then perform routing on the individual links that are bound to them thus achieving network separation.

[0006] A router generally comprises a routing element and a forwarding element. To implement virtual routing, the routing and forwarding elements have to be distinctly separate. There are multiple ways of achieving this separation.

BRIEF SUMMARY OF THE INVENTION

[0007] A routing device is disclosed. The routing device includes an operating system kernel, a virtual router residing external to the operating system kernel, a router manager configured to manage the virtual router, an application residing external to the virtual router, and a number of physical interfaces.

[0008] The application is able to selectively interact with the virtual router and the operating system kernel on a dynamic basis in order to have the virtual router and the operating system kernel perform a number of tasks for the application.

[0009] The virtual router further includes a routing protocol stack configured to handle a number of routing protocols, a number of interface drivers configured to communicate with corresponding physical interfaces, an Internet Protocol (IP) stack configured to interact with the routing protocol stack and perform a forwarding function via the interface drivers, the IP stack having a forwarding information table, information from which is used to perform the forwarding function; and a socket layer having a corresponding socket application programming interface, the socket layer configured to facilitate interactions between the IP stack and one or more socket applications, such as, the routing protocol stack and other external applications. The remote socket application programming interface is used to facilitate communications with the socket layer. Furthermore, the socket application programming interface allows multiple applications to interact with the IP stack via the socket layer. The IP stack of the virtual router resides external to the operating system kernel.

[0010] The operating system kernel further includes an associated socket layer, the socket layer having a corresponding socket application programming interface. The application is able to communicate with the operating system kernel via the associated socket layer using the corresponding socket application programming interface to have the operating system kernel perform one or more tasks.

[0011] In one exemplary implementation, the routing device of the present invention is incorporated into an UNIX system and software is used to implement the virtual router and the router manager.

[0012] Reference to the remaining portions of the specification, including the drawings and claims, will realize other features and advantages of the present invention. Further features and advantages of the present invention, as well as the structure and operation of various embodiments of the present invention, are described in detail below with respect to accompanying drawings, like reference numbers indicate identical or functionally similar elements.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] FIG. 1 is a simplified schematic diagram illustrating an exemplary embodiment of the present invention; and

[0014] FIG. 2 is a simplified schematic diagram further illustrating an exemplary embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0015] The present invention in the form of one or more exemplary embodiments will now be described. According to an exemplary aspect, the method of the present invention achieves separation of elements in a router by having each virtual router with its own IP (Internet Protocol) stack, routing protocol stack and forwarding information table. This is achieved by implementing an IP stack that runs in the user space on multi-user and multi-process system, such as, a UNIX system. In a typical IP network element, the IP stack is always part of the kernel. By running the IP stack in the user domain, it is possible to build multiple virtual routers on a single multi-process system. The forwarding information table is implemented in hardware. The network links are on the line cards that are associated with individual virtual routers. The routing protocols are also part of the virtual router process and exchange routing information over the network links and update the routing table on the line cards.

[0016] According to one exemplary implementation, the TCP/IP stack runs in the user space. The user space TCP/IP stack provides the standard socket interface to the routing protocols for updating the routing table, which facilitates easy porting of the routing protocols. As will be further described below, the standard socket interface can also be used

to allow an application that is external to the virtual router to communicate with the IP stack and have certain tasks performed via the virtual router.

[0017] In one implementation, the IP stack is not modified. Because the IP stack is not modified, it behaves exactly the same as any other IP stack. As a result, the porting of routing protocols or other applications that use sockets becomes easy. Moreover, by not changing the IP stack, there is no need to re-test the IP stack; otherwise, the IP stack would have to be tested, the testing of which is a long drawn and expensive process.

[0018] Having separate TCP/IP and routing protocol stacks for each virtual router requires a large amount of memory. To conserve memory, dynamic libraries of the TCP/IP and the routing protocol stacks are created. By using dynamic libraries, the UNIX operating system is directed to maintain a single copy of the stack libraries in the memory, thus reducing memory requirements.

[0019] FIG. 1 is a simplified schematic diagram illustrating one embodiment of the present invention. As shown in FIG. 1, a routing device 10 includes a number of physical interfaces 12, a number of virtual routers 14 and a router manager 16. The physical interfaces 12 are designed to be coupled or bound to corresponding packet sources to receive packets to be forwarded. Each virtual router 14 is designed to be coupled to one or more of the physical interfaces 12. Different virtual routers 14 can be coupled to different physical interfaces 14. The router manager 16 controls the coupling of the virtual router 14 to its corresponding physical interfaces 12. The coupling is performed on a dynamic basis. In other words, a virtual router 14 can be coupled to different physical interfaces 12 at different times as managed by the router manager 16. In one implementation, upon startup of the routing device 10, none of the virtual routers 14 is bound to any physical interfaces 12. The router manager 16 handles the subsequent coupling of the virtual routers 14 to their corresponding physical interfaces 12. The operator of the routing device 10 is responsible for creating these bindings between the physical interfaces 12 and the virtual routers 14.

[0020] Furthermore, the routing device 10 includes at least one application 18. The application 18 can be a process that runs on the underlying operating system of the routing device 10. The application 18 is external to the virtual routers 14 and is able to selectively communicate with the virtual routers 14 so as to have tasks performed, as will be further described below. Optionally, the application 18 is also able to communicate with the underlying operating system of the routing device 10 to have tasks performed.

[0021] FIG. 2 is a simplified schematic diagram further illustrating an embodiment of the routing device 20. As shown in FIG. 2, in this embodiment, the routing device 20 includes a number of virtual routers 22 and an operating system kernel 34. Each virtual router 22 includes a number of elements comprising a routing protocol stack, a socket layer 28, an IP stack 30 and a number of interface drivers 36. The routing protocol stack includes a number of entities 26 that are capable of respectively handling routing protocols supported by the virtual router 22 including, for example, RIP (Routing Information Protocol), OSPF (Open Shortest Path First), BGP (Border Gateway Protocol) and the like. Other routing protocols may be supported in the routing protocol stack depending on the requirements of the virtual router 22. Each entity 26 is associated with a particular routing protocol and contains an appropriate routing table.

[0022] As mentioned above, each virtual router 22 includes the socket layer 28. The socket layer 28 facilitates communications between the IP stack 30 and other elements, such as, the routing protocol stack and the application 24. Communications with the socket layer 28 are achieved using a socket application programming interface (API). In one exemplary implementation, the socket API follows the standard Berkeley socket API. Based on the disclosure and teachings provided herein, a person of ordinary skill in the art will know of other protocols or standards that can be used as the socket API in accordance with the present invention. The use of the socket layer 28 and its API will be further described below.

[0023] Each virtual router also includes the IP stack 30. The IP stack 30 contains a forwarding information base (FIB). Via the socket layer 28, the IP stack 30 interacts with the entities 26 and uses the FIB to facilitate forwarding of packets to their intended destinations; the IP stack 30 is also able to communicate with the application 24 to carry out one or more tasks for the application 24.

[0024] Virtual routers 22 may be configured in a similar or different manner depending on the routing design and/or requirements of the routing device 20. One virtual router 22a within the routing device 20 can be different from another virtual router 22b with respect to their constituent elements. For example, one virtual router 22a may have a routing protocol stack that supports one set of routing protocols, while the routing protocol stack of another virtual router 22b may support a different set of routing protocols; and one virtual router 22a may support one set of physical interfaces, while another virtual router 22b may support a different set of physical interfaces. Based on the disclosure and teachings provided

herein, a person of ordinary skill in the art will know of other ways and/or methods to configure a virtual router in accordance with the present invention.

[0025] Furthermore, having an IP stack in each virtual router 22 allows many applications that use the standard socket API to be run. The routing sockets use the standard socket API to update the FIB. Similarly, any application that require standard socket API can be run within the context of the virtual router process.

[0026] As the number of applications grows, the size of each virtual router process also increases thereby affecting performance. To overcome this performance issue, in one embodiment, the socket API is extended to support remote applications. This involves exporting the socket API to the remote application using an interprocess communication infrastructure. The applications then use the remote socket API as if they were directly connected to the socket layer sub-system. Each of the socket calls made by the remote applications is translated into remote procedure calls by the underlying socket library.

[0027] The remote socket library is flexible enough to support applications that are either internal or external. The library that is linked controls how the socket calls are to be handled. This implementation keeps in check the size of the virtual router and at the same time improves application performance. Apart from performance improvements, this design also offers flexibility to replace either virtual router software or the application software without affecting each other.

[0028] Under this implementation, the applications are made virtual router aware, meaning that an applications when using the socket API has to know the virtual router that it is communicating with.

[0029] In one exemplary aspect, as shown in FIG. 2, the application 24 is able to selectively communicate with the virtual routers 22 and the operating system kernel 34 to carry out its tasks. For example, the application 24 can communicate with the virtual router 22a via the corresponding socket layer 28 using the associated socket API. Optionally, the application 24 can also communicate with the operating system kernel 34 via a corresponding socket layer (not shown) using the associated socket API. Consequently, the application 24 can utilize combinations of virtual routers 22 and the operating system kernel 34 to complete its tasks. For example, if the application 24 is handling three (3) clients (not shown), the application 24 can use virtual routers 22a and 22b and the operating system kernel 34 to respectively execute tasks for the three (3) clients. It should be understood that there can be

any number of virtual routers 22 implemented within the routing device 20, depending on system and/or design requirements and constraints. Based on the disclosure and teachings provided herein, a person of ordinary skill in the art will know of other ways and/or methods to use the virtual routers and the operating system kernel to accommodate one or more applications in accordance with the present invention.

[0030] In one implementation, the present invention is implemented using the Solaris® system manufactured by Sun Microsystems. The Solaris® system is an UNIX operating system. In this implementation, the virtual routers 32 are created in the user space. Furthermore, one UNIX process is used per virtual router 32. Since the UNIX operating system is a time-shared system, it schedules the virtual router processes appropriately thereby minimizing any scheduling problem.

[0031] In an exemplary implementation, the present invention is implemented using software in the form of control logic, in either an integrated or a modular manner. Alternatively, hardware or a combination of software and hardware can also be used to implement the present invention. Based on the disclosure and teachings provided herein, a person of ordinary skill in the art will know of other ways and/or methods to implement the present invention.

[0032] It is understood that the examples and embodiments described herein are for illustrative purposes only and that various modifications or changes in light thereof will be suggested to persons skilled in the art and are to be included within the spirit and purview of this application and scope of the appended claims. All publications, patents, and patent applications cited herein are hereby incorporated by reference for all purposes in their entirety.